Key-Insulated Symmetric Key Cryptography and Mitigating Attacks against Cryptographic Cloud Software

Yevgeniy Dodis
Dept. of Computer Science
New York University
dodis@cs.nyu.edu

Weiliang Luo
Dept. of Computer Science
University of Texas at San Antonio
wluo@cs.utsa.edu

Shouhuai Xu
Dept. of Computer Science
University of Texas at San Antonio
shxu@cs.utsa.edu

Moti Yung
Google
moti@cs.columbia.edu

ABSTRACT

Software-based attacks (e.g., malware) pose a big threat to cryptographic software because they can compromise the associated cryptographic keys in their entirety. In this paper, we investigate key-insulated symmetric key cryptography, which can mitigate the damage caused by repeated attacks against cryptographic software. To illustrate the feasibility of key-insulated symmetric key cryptography, we also report a proof-of-concept implementation in the Kernel-based Virtual Machine (KVM) environment.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Security

Keywords

Key-insulation, key management, cloud computing

1. INTRODUCTION

The motivation of the present study is two-fold. From a system perspective, a cloud should facilitate secure communications between the applications that run in different Virtual Machines (VMs). A particularly devastating attack is that the attacker fully compromises the cryptographic keys associated with the cryptographic software. It is therefore imperative to mitigate the damage caused by such full (rather than partial) exposure of cryptographic keys. From a cryptographic perspective, the notion of key-insulated cryptography has been investigated but only in the public-key setting [7, 8]. While key-insulated public key cryptography may be adapted to fulfill the functions we aim to offer, key-insulated symmetric key cryptography can deal with bulk data more efficiently and can ease the task of key management in such a relatively centralized environment. Key-insulated symmetric key cryptography is interesting also for its own sake because a symmetric key can be exposed at the sender or receiver side; whereas an asymmetric (private) key can be exposed at the receiver side only.

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

Our contributions.

We present definition and construction of key-insulated symmetric key cryptography. We show how key-insulated symmetric key cryptography can be adopted to mitigate attacks against cryptographic cloud software, we consider its integration into Trusted Virtual Domain (TVD) [11].

Related work.

There are three approaches to mitigate the damage caused by the full compromise of cryptographic keys. The first is the *primitives approach*. One strategy is to divide the system time into periods, and change the cryptographic key frequently. An example is forward-security [1, 2, 3], which ensures that compromise of a key during one period does not allow the attacker to obtain the key in any past period. Another strategy is represented by the notion called threshold cryptosystems [4], which splits the key (rather than the system time) into multiple shares such that a key is not compromised until after a sufficient number of shares are compromised. The second is the *architectural approach*. In this approach a cryptographic key is protected in a tamper-resistant hardware [13]. The third is the *hybrid approach*. This approach has the advantages of the two approaches mentioned above. Two examples are key-insulated public key cryptosystems [7, 8] and intrusion-resilient public key cryptosystems [10, 5, 6]. The present paper follows this approach and investigates key-insulated symmetric key schemes.

2. KEY-INSULATED SYMMETRIC CRYPTOGRAPHY

2.1 Model and Definition

The lifetime of the system is divided into periods $1, \ldots, N$

(e.g., days). Each user/participant has a *device* (e.g., a hardware co-processor, or a trusted software module in VMM), and a networked *computer*. Denote by $\mathcal{P}$ the set of identities of the users (e.g., VMs), where $|\mathcal{P}| \geq 2$. At the beginning of time period t ($1 \leq t \leq N$), each user's computer obtains certain information from the user's device, derives a key SK_t for this time period (i.e., all the relevant users' computers obtain the same SK_t). A networked computer is subject to repeated compromise. Although it is assumed to be harder, a device may also be compromised. Our model and definition of key-insulated symmetric key schemes are adapted from the ones of key-insulated public key cryptography [7, 8], but here are some fundamental differences between the public-key setting and the secret-key setting.

DEFINITION 1. *A* key-updating symmetric key scheme Π *is a 5-tuple of polynomial-time algorithms* $(\mathcal{G}, \mathcal{U}^*, \mathcal{U}, \mathcal{E}, \mathcal{D})$*:*

- $\mathcal{G}$*, the probabilistic* key generation algorithm *that takes as input a security parameter* k *and the total number of time periods* N*. It returns* $\mathsf{SK}^{(Dev)} \stackrel{def}{=} \{\mathsf{SK}^{(\mathsf{ID}.Dev)}\}_{\mathsf{ID}\in\mathcal{P}}$ *and* $\mathsf{SK}^{(Comp)} \stackrel{def}{=} \{\mathsf{SK}^{(\mathsf{ID}.Comp)}\}_{\mathsf{ID}\in\mathcal{P}}$*, where* $\mathsf{SK}^{(\mathsf{ID}.Dev)}$ *and* $\mathsf{SK}^{(\mathsf{ID}.Comp)}$ *are user* ID*'s device master key and computer master key, respectively.*
- $\mathcal{U}^*$*, the* device key-update algorithm *that takes as input an index* t*, where* $1 \leq t \leq N$*, and the device master key* $\mathsf{SK}^{(\mathsf{ID}.Dev)}$*. It returns a partial secret key* $\mathsf{SK}_t^{\mathsf{ID}.Dev}$*.*
- $\mathcal{U}$*, the* computer key-update algorithm *that takes as input an index* t*, the secret key* SK_{t-1} *for time period* $t-1$ *(where* $\mathsf{SK}_0 = \perp$*), the partial secret key* $\mathsf{SK}_t^{\mathsf{ID}.Dev}$*, and the computer master key* $\mathsf{SK}^{(\mathsf{ID}.Comp)}$*. It returns a secret key* SK_t *for time period* t*, and erases* SK_{t-1} *as well as* $\mathsf{SK}_t^{\mathsf{ID}.Dev}$*.*
- $\mathcal{E}$*, the* encryption algorithm *that takes as input an index* t*, a message* M*, and secret key* SK_t*. It returns a ciphertext* $\langle t, C\rangle$*.*
- $\mathcal{D}$*, the* decryption algorithm *that takes as input secret key* SK_t *and ciphertext* $\langle t, C\rangle$*. It returns the corresponding message* M *if the ciphertext is legitimate, and* $\perp$ *otherwise.*

For correctness, we require that for every message M and $1 \leq t \leq N$, it holds that $\Pr[\mathcal{D}_{\mathsf{SK}_t}(\mathcal{E}_{\mathsf{SK}_t}(t, M)) = M] = 1$.

2.2 Security Definition

We consider three types of exposures: (1) ordinary key exposure, which models the (repeated) compromise of ID's computer and leaks SK_t and $\mathsf{SK}^{(\mathsf{ID}.Comp)}$; (2) key-update exposure, which models the (repeated) compromise of ID's computer during the key-updating step and leaks SK_{t-1}, SK_t, and $\mathsf{SK}^{(\mathsf{ID}.Comp)}$); and (3) device master key exposure, which models the compromise of ID's device and leaks $\mathsf{SK}^{(\mathsf{ID}.Dev)}$. Formally, we give the adversary access to three (possibly five) types of oracles.

- Key exposure oracle $\mathsf{Exp}_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot,\cdot)$: It, on input $t \in \{1,\ldots,N\}$ and $\mathsf{ID} \in \mathcal{P}$, returns the period secret key SK_t.
- Key exposure oracle $\mathsf{Dev}(\cdot,\cdot)$: It, on input $t \in \{1,\ldots,N\}$ and $\mathsf{ID} \in \mathcal{P}$, returns $SK^{(\mathsf{ID}.Dev)}$.
- Left-or-right encryption oracle $\mathsf{LR}_{\mathcal{E},\vec{b}}(\cdot,\cdot,\cdot)$: It is defined as $\mathsf{LR}_{\mathcal{E},\vec{b}}(t, M_0, M_1) \stackrel{def}{=} \mathcal{E}_{\mathsf{SK}_t}(t, M_{b_t})$, where $\vec{b} = b_1,\ldots,b_N \in \{0,1\}^N$. It models encryption requests by the adversary on (period, message) pairs.
- We may allow the adversary to have access to encryption oracle $\mathcal{E}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot,\cdot)$ that, on input t and M, computes and returns $\langle t, C\rangle \stackrel{def}{=} \mathcal{E}_{\mathsf{SK}_t}(t, M)$. This models a chosen-plaintext attack by the adversary.
- We may also allow the adversary to have access to decryption oracle $\mathcal{D}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot)$ that, on input $\langle t, C\rangle$, computes and returns $\mathcal{D}_{\mathsf{SK}_t}(\langle t, C\rangle)$. This models a chosen-ciphertext attack by the adversary.

We allow the adversary to interleave encryption requests and key exposure requests. Moreover, key exposure requests may be made adaptively and in any order.

DEFINITION 2. *For* $\mathsf{ID} \in \mathcal{P}$*, define* $\mathcal{T}'_{\mathsf{ID}} = \{t | 1 \leq t \leq N \wedge \exists \mathsf{Exp}_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(t, \mathsf{ID})\}$*, which corresponds to the points in time that the computer is compromised, and* $\mathcal{T}''_{\mathsf{ID}} = \{t | 1 \leq t \leq N \wedge \exists \mathsf{Dev}(t, \mathsf{ID})\}$*, which corresponds to the points in time that the device is compromised. Define* $t_{\mathsf{ID}} = \max(t'_{\mathsf{ID}}, t''_{\mathsf{ID}})$*, where* t'_{ID} *and* t''_{ID} *are defined as follows:*

- *If* $\mathcal{T}'_{\mathsf{ID}} \neq \emptyset$*, define* $t'_{\mathsf{ID}} = t$ *such that* $t \in \mathcal{T}'_{\mathsf{ID}}$ *and* $\forall\ t' \in \mathcal{T}'_{\mathsf{ID}}$*,* $t' \geq t$ *(i.e.,* t'_{ID} *is the first time that* ID*'s computer is compromised); otherwise, define* $t'_{\mathsf{ID}} = \infty$*.*
- *If* $\mathcal{T}''_{\mathsf{ID}} \neq \emptyset$*, define* $t''_{\mathsf{ID}} = t$ *such that* $t \in \mathcal{T}''_{\mathsf{ID}}$ *and* $\forall\ t'' \in \mathcal{T}''_{\mathsf{ID}}$*,* $t'' \geq t$ *(i.e.,* t''_{ID} *is the first time that* ID*'s device is compromised); otherwise, define* $t''_{\mathsf{ID}} = \infty$*.*

Note that t_{ID} *is the earliest point in time that* ID*'s computer and device have been compromised. Define* $\mathcal{T}_{\mathcal{P}} = \{t_{\mathsf{ID}} | \mathsf{ID} \in \mathcal{P}\}$*. Define* $t_\Pi = t$ *such that* $t \in \mathcal{T}_{\mathcal{P}}$ *and* $\forall\ \mathsf{ID} \in \mathcal{P}$*,* $t \leq t_{\mathsf{ID}}$*. Note that* t_Π *is the earliest point in time that* ID*'s computer and device have been compromised, which means that all of the cryptographic keys have been compromised. We say that* Π *is* never-compromised *if* $t_\Pi = \infty$*, and* Π *is* compromised *at time* t_Π *otherwise. Define* $\mathcal{T}'_{\mathcal{P}} = \bigcup_{\mathsf{ID}\in\mathcal{P}} \mathcal{T}'_{\mathsf{ID}}$*. We say that a never-compromised* Π *is* un-exposed *at time period* t *if* $t \notin \mathcal{T}'_{\mathcal{P}}$*.*

For a *never-compromised* Π, we require *key-insulation* specified below; for a Π that is *compromised* at some time t_Π, we require *augmented key-insulation* specified below. Informally, Π is *key-insulated* if the probability that any probabilistic polynomial-time adversary succeeds in guessing the value of b_t for any un-exposed time period t is negligibly more than 1/2. More formally,

DEFINITION 3. (key-insulation) *Let* Π *be a key-updating symmetric key scheme. For adversary* $\mathcal{A}$*, define:*

$$\mathsf{Succ}_{A,\Pi}(k) \stackrel{def}{=} \Pr\left[\begin{array}{l}(\mathsf{SK}^{(Dev)}, \mathsf{SK}^{(Comp)}) \leftarrow \mathcal{G}(1^k, N);\\ \vec{b} \leftarrow \{0,1\}^N;\\ (t, b) \leftarrow \\ A^{\mathsf{LR}_{\mathcal{E},\vec{b}}(\cdot,\cdot,\cdot),\mathsf{Exp}_{\mathsf{SK}(Dev),\mathsf{SK}(Comp)}(\cdot,\cdot),\mathsf{Dev}(\cdot,\cdot),\mathcal{O}_1(\cdot,\cdot),\mathcal{O}_2(\cdot)}() :\\ b = b_t\end{array}\right],$$

where $\mathcal{O}_1(\cdot,\cdot) = \perp$ *for known-plaintext attack and* $\mathcal{O}_1(\cdot,\cdot) = \mathcal{E}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot,\cdot)$ *for chosen-plaintext attack, and* $\mathcal{O}_2(\cdot) = \perp$

meaning that the adversary has no access to the decryption oracle and $\mathcal{O}_2(\cdot) = \mathcal{D}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot)$ *meaning that the adversary has access to the decryption oracle (i.e., chosen-ciphertext attack in which case the adversary is not allowed to query* $\mathcal{D}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\langle t, C\rangle)$ *if* $\langle t, C\rangle$ *was returned by* $\mathsf{LR}_{\mathcal{E},\vec{b}}(t,\cdot,\cdot)$*). Then,* Π *is* (T, N)*-key-insulated if for any probabilistic polynomial-time* $\mathcal{A}$ *such that* $t_\Pi = \infty$ *(i.e.,* Π *is* never-compromised*),* $t \notin \mathcal{T}'_{\mathcal{P}}$ *(i.e.* Π *is* un-exposed *at period* t*), and* $|\mathcal{T}'_{\mathcal{P}}| \leq T$*,* $|\mathsf{Succ}_{A,\Pi}(k) - 1/2|$ *is negligible. Moreover, we say a* $(N-1, N)$*-key-insulated symmetric key scheme achieves* optimal key-insulation.

For the notion of key-insulation, it may be desirable to consider an extra property called *secure key updates* below. We call the following attack a key-update exposure at period t on ID's computer: an adversary breaks into user ID's computer while a key update is taking place (i.e., the exposure occurs between two periods $t-1$ and t). In this case, the adversary receives SK_{t-1}, $\mathsf{SK}_t^{\mathsf{ID}.Dev}$, $\mathsf{SK}^{(\mathsf{ID}.Comp)}$, and (can compute) SK_t. Informally, we say a scheme has *secure key updates* if a key-update exposure at period t on ID's computer is equivalent to key exposures at periods $t-1$ and t on ID's computer and no more. More formally:

DEFINITION 4. (secure key updates for key-insulation) *A key-updating symmetric key scheme* Π *has* secure key updates *if the view of any adversary* $\mathcal{A}$ *making a key-update exposure request at time period* t *on* ID*'s computer can be perfectly simulated by an adversary* $\mathcal{A}'$ *who makes key exposure requests at periods* $t-1$ *and* t *on* ID*'s computer.*

Informally, we say Π is augmented key-insulated if the probability that any polynomial-time adversary succeeds in guessing the bit b_t corresponding to $\mathsf{LR}_{\mathcal{E},\vec{b}}(t, M_1, M_2)$ is negligibly more than $1/2$, where $t < t_\Pi$. More formally,

DEFINITION 5. (augmented key-insulation) *Let* Π *be a key-updating symmetric key encryption scheme. For adversary* $\mathcal{A}$*, define the following:*

$$\mathsf{Succ}_{I,\Pi}(k) \stackrel{def}{=} \Pr\left[\begin{array}{l}(\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}) \leftarrow \mathcal{G}(1^k, N);\\ \vec{b} \leftarrow \{0,1\}^N;\\ (t_\Pi, t, b) \leftarrow\\ I^{\mathsf{LR}_{\mathcal{E},\vec{b}}(\cdot,\cdot,\cdot),\mathsf{Exp}_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot,\cdot),\mathsf{Dev}(\cdot,\cdot),\mathcal{O}_1(\cdot,\cdot),\mathcal{O}_2(\cdot)}() :\\ b = b_t\end{array}\right],$$

where $t < t_\Pi$ *and* $t \notin \mathcal{T}'_{\mathcal{P}}$*,* $\mathcal{O}_1(\cdot,\cdot) = \perp$ *for known-plaintext attack and* $\mathcal{O}_1(\cdot,\cdot) = \mathcal{E}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot,\cdot)$ *for chosen-plaintext attack, and* $\mathcal{O}_2(\cdot) = \perp$ *meaning that the adversary has no access to the decryption oracle and* $\mathcal{O}_2(\cdot) = \mathcal{D}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\cdot)$ *meaning that the adversary has access to the decryption oracle (i.e., chosen-ciphertext attack in which case the adversary is not allowed to query* $\mathcal{D}^*_{\mathsf{SK}^{(Dev)},\mathsf{SK}^{(Comp)}}(\langle t, C\rangle)$ *if* $\langle t, C\rangle$ *was returned by* $\mathsf{LR}_{\mathcal{E},\vec{b}}(t,\cdot,\cdot)$*).* Π *is* augmented key-insulated *if: (1) when* $t_\Pi < \infty$*,* $|\mathsf{Succ}_{I,\Pi}(k) - 1/2|$ *is negligible for any probabilistic polynomial-time algorithm* $\mathcal{A}$*, and (2) when* $t_\Pi = \infty$*,* Π *is key-insulated.*

2.3 Key-Insulated Symmetric Key Scheme

Let (G, E, D) be a secure symmetric key cryptosystem, where G is the key generation algorithm which takes as input a security parameter k and outputs a key K, $E_K(\cdot)$ is encryption algorithm, and $D_K(\cdot)$ is the decryption algorithm. We refer to [12] for its security definitions. Let $\{f_K\} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$ be a pseudorandom function family keyed by $K \in \{0,1\}^k$ [9]. The $(N-1, N)$-key-insulated symmetric key scheme Π for two party communication (i.e., $|\mathcal{P}| = 2$) is specified as follows.

- Key Generation. This algorithm is executed in a secure environment. Suppose $\{x_i\}_{1\leq i\leq 4}$ are uniformly chosen from $\{0,1\}^k$. Alice stores (x_1, x_2) on her computer, and (x_3, x_4) on her device; Bob stores (x_1, x_3) on his computer, and (x_2, x_4) on his device.
- Device Key-Update. At the beginning of period t ($1 \leq t \leq N$), Alice's device sends $f_{x_3}(t) \oplus f_{x_4}(t)$ to her computer, and Bob's device sends $f_{x_2}(t) \oplus f_{x_4}(t)$ to his computer.
- Computer Key-Update. The secret key for period t is $\mathsf{SK}_t = f_{x_1}(t) \oplus f_{x_2}(t) \oplus f_{x_3}(t) \oplus f_{x_4}(t)$, which can be derived by Alice's computer and Bob's computer.
- Encryption. For period t, set $\mathcal{E}_{\mathsf{SK}_t}(t, M) = E_{\mathsf{SK}_t}(M)$.
- Decryption. For period t, set $\mathcal{D}_{\mathsf{SK}_t}(\langle t, C\rangle) = D_{\mathsf{SK}_t}(C)$.

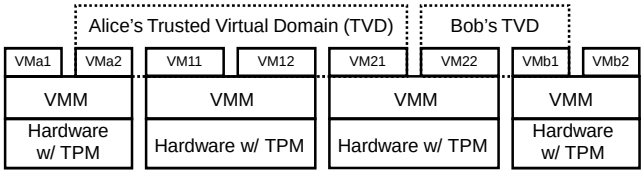
THEOREM 1. *Suppose* (G, E, D) *is a secure symmetric key encryption scheme, and* $\{f_K\}$ *is a secure pseudorandom function family. Then,* Π *is* $(N-1, N)$*-key-insulated with secure key updates.*

2.4 Integrating Key-Insulated Scheme with TVD

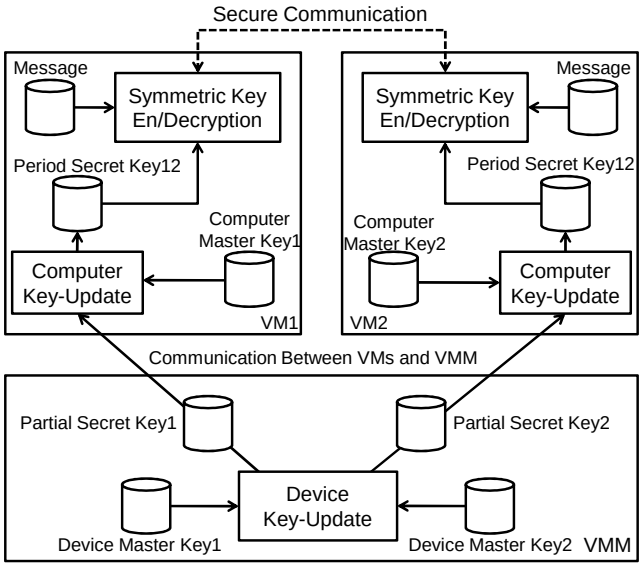
As illustrated in Figure 1(a), a TVD allows a customer (Alice) to use multiple VMs running on top of multiple physical computers in the cloud. The communications between the applications running in the same TVD should be protected from the environment outside the TVD.

As illustrated in Figure 1(b), where we consider two VMs running on top of the same Virtual Machine Monitor (VMM) for the sake of simplicity, key-insulated symmetric key cryptography can mitigate the repeated exposures of secret keys. More specifically, we can let each VM hold a master key (called *computer master key*), and let the VMM hold a set of master keys (called *device master keys*). At the beginning of each time period, a VM receives from the *device key-update* software module a *partial secret key*, which is derived from the device master key. The *computer key-update* module will derive a *period secret key* from the partial secret key and the computer master key. The period secret key is the symmetric key for protecting the communications between the two VMs that belong to the same TVD. As a proof of concept, we report our implementation of key-insulated symmetric key scheme in the KVM environment as well as its performance measurements. Since the difference between standard symmetric key cryptography and key-insulated symmetric key cryptography is the key update operation at the beginning of each time period, the performance metric we consider is the key update time, which is dependent upon the number of VMs one will communicate with, and is dependent upon the number of VMs running on top of a single physical computer.

As a proof of concept, we implemented the key-insulated symmetric key scheme in the KVM environment. Our experimental system was a desktop computer. The hardware was two x86 processors at 2.5 GHz with 2GB memory. The

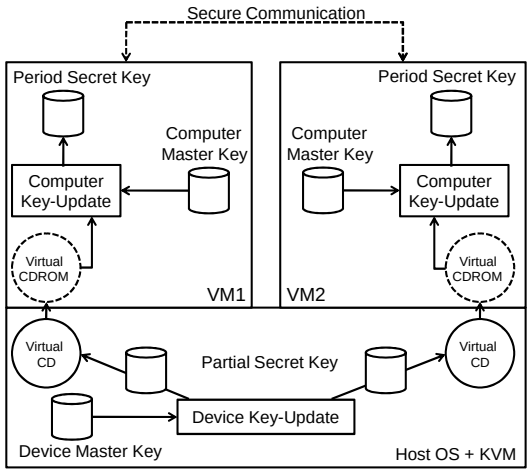


(a) TVD in cloud environment

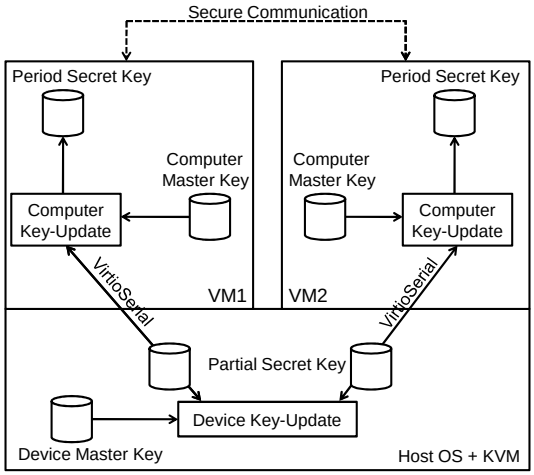


(b) TVD with key-insulated symmetric key encryption

Figure 1: Key-insulated symmetric key cryptography and TVD



(a) Implementation approach I



(b) Implementation approach II

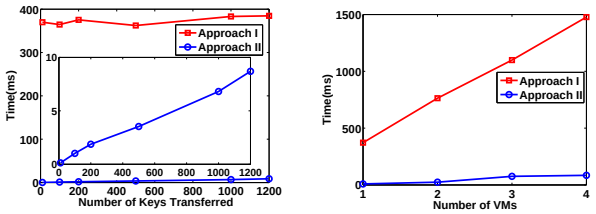
Figure 2: Two approaches for implementing key-insulated symmetric key schemes

Host OS was Ubuntu 11.10. The guest OS was Ubuntu 10.04. The *device* in our formal model was implemented as a small software module in KVM, called Device Key-Update, which implements the Device Key-Update algorithm. Similarly, we implemented a Computer Key-Update module in the VM. We choose to implement the *device* in KVM because we can modify the source code. Note that TPM does not allow one to run any third-party code.

There are two approaches to realize key-insulation in KVM. The difference between the two approaches is how the Device Key-Update module and the Computer Key-Update module communicate. Figure 2(a) demonstrates approach I, which utilizes the virtual CDROM mechanism. Specifically, the Device Key-Update module in KVM will write the key updates to a virtual CD (in the format of ISO file), and then "insert" the virtual CD into the CDROM device of the respective VM. Figure 2(b) demonstrates approach II, which utilizes KVM's VirtioSerial feature that further allows the Computer Key-Update module to acknowledge the receiving of key updates from the Device Key-Update module.

Since the secure communications between VMs using the period secret keys are the same as the standard use of symmetric key schemes, we want to demonstrate that the key update operations do not incur any significant performance cost. This is justified by the fact that the cost for evaluating pseudorandom functions, for which we used AES-128, can be almost ignored in practice. The most significant part of the cost is the communication from the Device Key-Update module to the Computer Key-Update module. Since one VM may need to conduct secure communications with multiple or many other VMs, we measure the performance impact of the number of key updates (i.e., the number of VMs with which one VM communicates). Since a VMM needs to support multiple VMs simultaneously, we measure the performance impact of the number of VMs running on top of a physical machine.

Figure 3(a) compares the communication costs of the two approaches with respect to the number of key updates. In the experiments, we ran a single VM on top of KVM. Suppose one VM needs to conduct secure communications with up to 1,200 other VMs, which is possible with the TVD abstraction mentioned in the Introduction, the Computer Key-Update module in the VM needs to receive up to 1,200 key updates from the Device Key-Update module in the KVM. It is clear that Approach II is two orders of magnitude faster than Approach I. Because Approach II incurs very small communication cost, we also plotted the zoomed-in version of the curve. It is interesting to note that the communication cost of Approach I is roughly independent of the number of key updates; whereas, the communication cost of Ap-



(a) Impact of number of keys (b) Impact of number of VMs

Figure 3: Performance evaluation

proach II is proportional to the number of key updates. This phenomenon is inherent to the communication mechanisms. Figure 3(b) compares the communication costs of the two approaches with respect to the number of VMs running on top of a single KVM. In our experiments, we ran 1, 2, 3, 4 VMs on the aforementioned desktop hardware platform, respectively. In any case, each VM was allocated with 256MB memory and ran Ubuntu 10.04. The curves correspond to that each VM receives 1,200 key updates from the Device Key-Update module. In either case, we observe that the communication cost is roughly proportional to the number of VMs running on the hardware platform. In summary, we observe that Approach II is much more efficient than Approach I.

2.5 Augmented Key-Insulation Scheme

Augmented key-insulated symmetric key scheme offers a stronger security guarantee under certain circumstances. Its deployment and device-to-computer communication cost are essentially the same as the ones of the above key-insulated symmetric key scheme, except that the key update algorithms need to evaluate two more pseudorandom functions (e.g., AES-128). This explains why we do not repeat the implementation part. Let (G, E, D) be a secure symmetric key cryptosystem. The augmented key-insulated symmetric key scheme for secure two party communication is specified below. A key chain specified by $X_{i,0}$ is defined as $X_{i,t} = f_{X_{i,t-1}}(0)$ for $1 \le t \le N$.

- Key Generation. This algorithm is executed in a secure environment. Suppose $\{X_{i,0}\}_{1 \le i \le 4}$ is a set of secrets uniformly chosen from $\{0,1\}^k$. Alice stores $(X_{1,0}, X_{2,0})$ on her computer, and $(X_{3,0}, X_{4,0})$ on her device; Bob stores $(X_{1,0}, X_{3,0})$ on his computer, and $(X_{2,0}, X_{4,0})$ on his device.

- Device Key-Update. At the beginning of time period t $(1 \le t \le N)$, Alice's device holds $(X_{3,t-1}, X_{4,t-1})$, and Bob's device holds $(X_{2,t-1}, X_{4,t-1})$. This algorithm includes the following steps.

  1. Alice's device sends $f_{X_{3,t-1}}(1) \oplus f_{X_{4,t-1}}(1)$ to her computer; Bob's device sends $f_{X_{2,t-1}}(1) \oplus f_{X_{4,t-1}}(1)$ to his computer.
  2. Alice's device computes and holds $(X_{3,t}, X_{4,t})$ and erases $(X_{3,t-1}, X_{4,t-1})$; Bob's device computes and holds $(X_{2,t}, X_{4,t})$ and erases $(X_{2,t-1}, X_{4,t-1})$.

- Computer Key-Update. At the beginning of period t, where $1 \le t \le N$, Alice's computer holds secrets $(\mathsf{SK}_{t-1}; X_{1,t-1}, X_{2,t-1})$; Bob's computer holds secrets $(\mathsf{SK}_{t-1}; X_{1,t-1}, X_{3,t-1})$.

  1. Both Alice's computer and Bob's computer compute and hold $\mathsf{SK}_t = f_{X_{1,t-1}}(1) \oplus f_{X_{2,t-1}}(1) \oplus f_{X_{3,t-1}}(1) \oplus f_{X_{4,t-1}}(1)$, which is the secret key for time period t.
  2. Alice's computer computes and holds the pair of secrets $(X_{1,t}, X_{2,t})$, erases $(\mathsf{SK}_{t-1}; X_{1,t-1}, X_{2,t-1})$; Bob's computer computes and holds the pair of secrets $(X_{1,t}, X_{3,t})$, erases $(\mathsf{SK}_{t-1}; X_{1,t-1}, X_{3,t-1})$. Recall that $X_{i,t} = f_{X_{i,t-1}}(0)$.

- Encryption. For period t, set $\mathcal{E}_{\mathsf{SK}_t}(t, M) = E_{\mathsf{SK}_t}(M)$.

- Decryption. for period t, set $\mathcal{D}_{\mathsf{SK}_t}(\langle t, C \rangle) = D_{\mathsf{SK}_t}(C)$.

THEOREM 2. *If (G, E, D) is a secure symmetric encryption scheme and $\{f_K\}$ is a secure pseudorandom function family, then* Π *is an augmented key-insulated symmetric key scheme.*

3. CONCLUSION

We presented the definition and constructions of key-insulated symmetric key schemes, and reported an implementation in the KVM environment.

Acknowledgement

We thank Jonathan Katz for discussions and suggestions.

4. REFERENCES

[1] R. Anderson, Invited Lecture, ACM CCS'97.

[2] M. Bellare and S. Miner, A Forward-Secure Digital Signature Scheme, Crypto'99.

[3] M. Bellare and B. Yee. Forward-Security in Private-Key Cryptography. RSA-CT'03.

[4] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. CRYPTO'89, pp 307-315.

[5] Y. Dodis, M. Franklin, J. Katz, A. Miyajo, and M. Yung. Intrusion-Resilient Public-Key Encryption. RSA-CT'03.

[6] Y. Dodis, M. Franklin, J. Katz, A. Miyajo, and M. Yung. A Generic Construction for Intrusion-Resilient Public-Key Encryption. RSA-CT'04.

[7] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public Key Cryptosystems. Eurocrypt'02.

[8] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Signature Schemes. PKC'03.

[9] O. Goldreich, S. Goldwasser, and S. Micali, How to Construct Random Functions, J. ACM, Vol. 33, No. 4, 1986, pp 210-217.

[10] G. Itkis and L. Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. Crypto'02.

[11] J. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn and R. C-aceres, Trusted Virtual Domains: Toward Secure Distributed Services, Proc. 2005 IEEE Workshop on Hot Topics in System Dependability.

[12] J. Katz and M. Yung, Complete Characterization of Security Notions for Probabilistic Private-Key Encryption, STOC'00.

[13] B. Yee. Using secure coprocessors. PhD thesis, Carnegie Mellon University, 1994.